oTree Manager:
Multi-user oTree installations made easy

Christian König-Kersting*
Heidelberg University

5 November 2018

Abstract:

oTree Manager is a software package designed to support the setup and management of multiple, production-ready oTree installations on the same server. Running in Docker containers, the individual instances run completely independent of each other while being less resource-intensive than traditional virtual machine setups. A convenient web interface provides both experimenters and laboratory managers with easy access to the most common actions and eliminates the need for command line interaction. Finally, oTree Manager comes with a novel *Lobby* feature which makes laboratory experiments that use oTree's rooms feature more convenient to run.

JEL: C88
Keywords: oTree, experimental economics, software, laboratory experiments

* Corresponding author: Alfred-Weber-Institute for Economics, Heidelberg University, Bergheimer Strasse 58, 69115 Heidelberg, Germany; Phone: +49 6221 54 2940; email: koenig.kersting@gmail.com.

1. Introduction

The oTree software package is increasingly used for conducting experiments both in the laboratory as well as online (Chen et al. 2016). A common problem for laboratory managers when preparing their laboratory for running experiments programmed using oTree is how to handle multiple experimenters. oTree itself does not come with built-in multi-user support.[1] As such, it assumes that only a single experimenter uses an installation at any given time. While experimenters can technically share a single installation by using the same web interface credentials, this can lead to a multitude of problems: Experimental sessions might be interrupted and data might be lost if one experimenter installs their experiments while the other one is running a session. Resetting the database to prepare for a newly added experiment might delete data the other experimenter has already collected, but not yet downloaded. Being defined installation-wide rather than per experiment, experimenters also cannot run experiments with different currency-, language-, or experimental currency settings simultaneously. In short, experimenters might unintendedly affect each other's experiments negatively.

The official oTree documentation provides a rough outline of how to better handle these situations. It boils down to giving each experimenter their own oTree installation. This can be done either by manually setting up an individual environment for each experimenter or by running individual Docker containers.[2] The first option involves manually creating virtual environments, configuring PostgreSQL and Redis databases, and handling port allocations, which is a tedious and error-prone effort, even for skilled laboratory managers. While coming with pre-configured databases and oTree installations, the second option still requires manual configuration for each additional experimenter. By default, both pathways yield inconvenient URLs, which require experimenters to address the correct ports to connect to their oTree installation. Clearly, these solutions quickly become very time-consuming, especially if oTree instances need to be added or removed on a regular basis, for example, when the number of experimentalists using oTree is growing or turnover is high.[3]

This paper presents *oTree Manager*, which is designed specifically to address these issues by automating the necessary steps to give each experimenter their own oTree instance[4]. oTree

[1] There is a glossary of technical terms in the appendix. It also provides links to the software packages mentioned throughout the article to avoid cluttering the footnotes.
[2] The community resources also include downloadable virtual machine templates and management scripts created by Felix Albrecht and Holger Gerhardt (https://otree-virtual-machine-manager.readthedocs.io).
[3] A common scenario are students who need individual instances to deploy their own experiments for course or thesis work.
[4] I use installation and instance synonymously.

Manager gives laboratory managers and experimenters an intuitive graphical user interface which allows them to set up and manage multiple, independent oTree instances on the same server without involving any manual configuration. The user interface is web-based and built on the same strong foundations as oTree itself. It is accessible by both laboratory managers and experimenters. Laboratory managers can create oTree installations and assign them to experimenters. These, in turn, can log in to easily set the oTree web interface password, reset the databases, or restart their oTree instance.

The development of oTree Manager adheres to three main objectives: 1) The final software should be easy to use for lab managers and experimenters alike. This includes that users should have to interact with the command line as little as possible. 2) It should be highly reliable and as fault tolerant as possible. In this regard, it is especially important that instances cannot (negatively) affect each other.[5] 3) The software should be easy to update and maintain and should not require extensive programming knowledge to setup. The overarching main goal is to make the lives of lab managers and experimenters easier. For a live demonstration, please visit https://demo.otree-manager.org.

The remainder of the paper is structured as follows: Section 2 presents the features oTree Manager offers to experimenters and laboratory managers. Section 3 explains oTree Manager's architecture and touches on its dependencies. Section 4 concludes the paper.

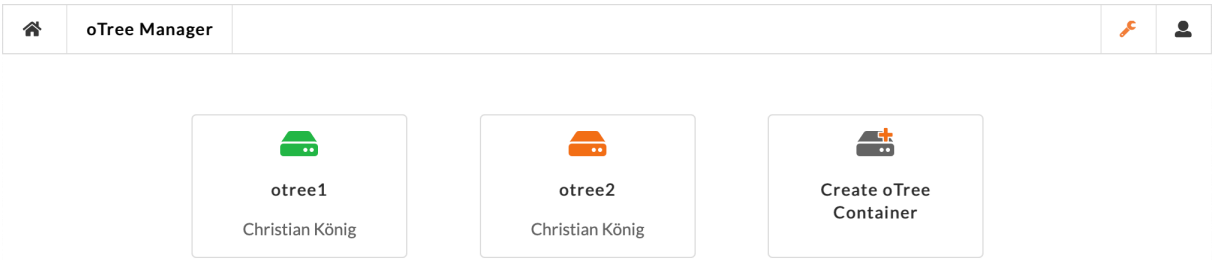
2. Features

2.1. Experimenters

Experimenters can comfortably manage their oTree instance using an intuitive web interface (Figure 1), to which they receive individual user accounts. oTree Manager comes with full support for standard user credential management: Users can change their password or request a new one should they have forgotten their old one. They can also set and reset their deployment keys, which allows for secure, password-less transfers of the experiments to the server. In oTree Manager, oTree instances are always associated with exactly one experimenter account (but experimenters can have multiple oTree instances). Experimenters can only see and configure their own instances, adhering to the principle of limiting access to only what is strictly necessary.

[5] It is especially important that errors in or outright crashes of one instance must not affect other instances.

Figure 1: oTree Manager Dashboard



Notes: This shows the first screen that (super-)users see after logging in. Quick navigation icons are located in the top right (wrench is super-user only). Instances which the current user has access to are shown in a grid below. Each instance is represented by a box, showing a status icon (green = ready to run experiments: orange = ready to upload experiments), its name as well as the assigned experimenter's name. Super-users also have the option to set up a new oTree instance.

oTree instances set up by oTree Manager are pre-configured for immediate use. They come with production-grade databases (PostgreSQL and Redis), oTree's authorization features are enabled, strong admin passwords are set and debug mode is turned off. Experimenters do not need to configure these aspects manually. The instances come with Git access for easy deployment of experimenter's code. Experimenters can add the repository as a new remote (just as they would for a cloud-hosted server on Heroku etc.) and simply push their latest version to the server. Instances managed by oTree Manager automatically detect newly pushed code revisions, install all required dependencies, and restart the oTree server.

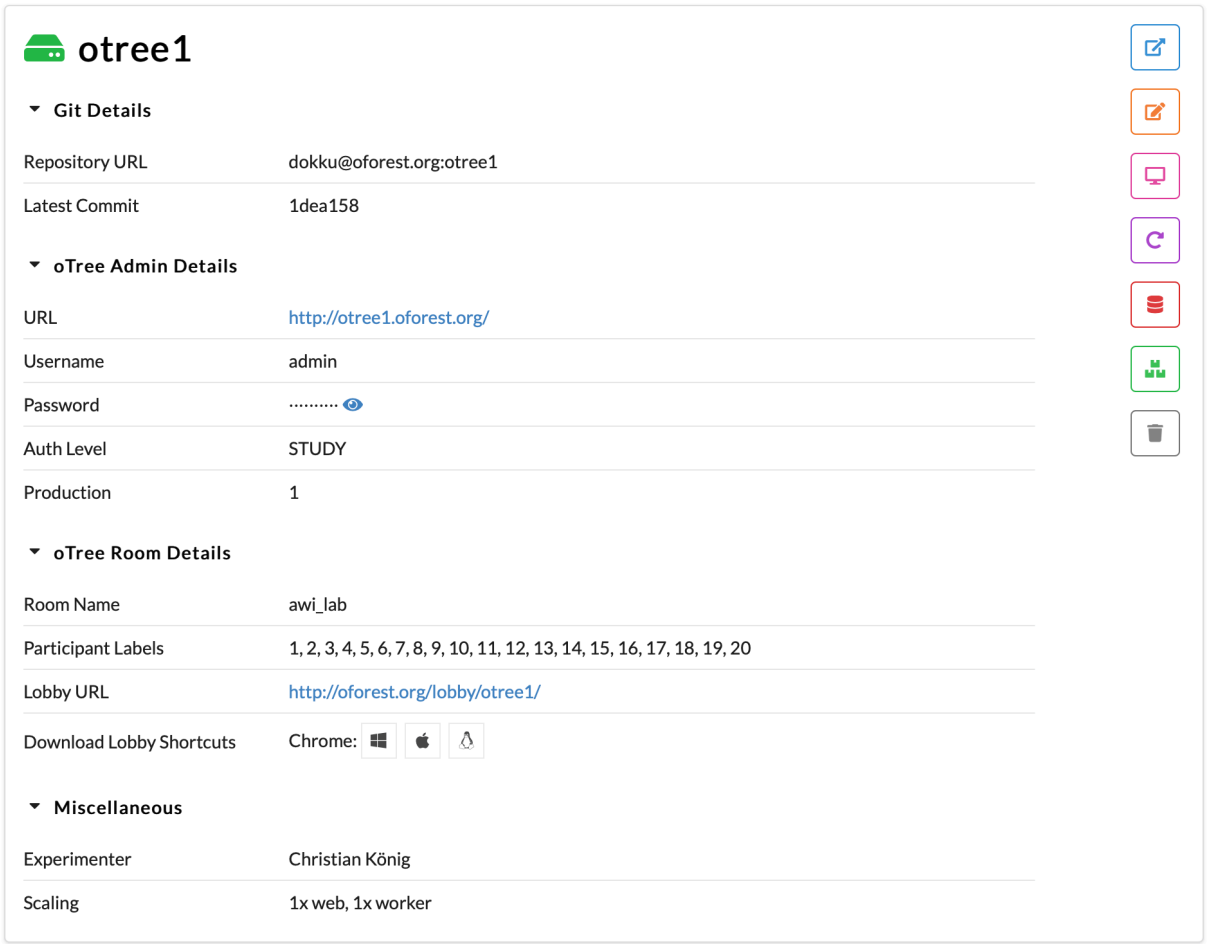
oTree Manager comes with a set of management features for experimenters: With the click of a button, they can (re-)set the password of oTree's web interface, restart the webserver, and reset the database (Figure 2). All of these actions would traditionally require command line interaction. Furthermore, they can integrate oTree's rooms feature with oTree Manager. If they do, oTree Manager automatically sets up a *Lobby* which can be opened on all client computers before the experimental session begins. It allows participants to signal that they are ready for the experiment to begin by clicking a button after having taken their seats (Figure 3). This step ensures that only those client computers show up on oTree's room management page, which actually have participants sitting in front of them. With this feature, experimenters can simply start all clients in the laboratory and direct the browsers to the Lobby's URL, irrespective of how many participants actually show up for the session. They do not need to worry about closing browser windows on unused client computers before starting the session.[6] oTree Manager provides desktop shortcuts for download, which start the Google Chrome (or

[6] There might be fewer participants present then client computers turned on due to participants not showing up to the session unexcused, for example. If the unused computers were pointed directly at oTree's room waiting page for session creation, the browsers would have to be closed manually in order not to have an incorrect number of clients connected when starting the actual experimental session.

Chromium) browser in kiosk mode and direct it to the lobby page for each participant label set up in oTree.

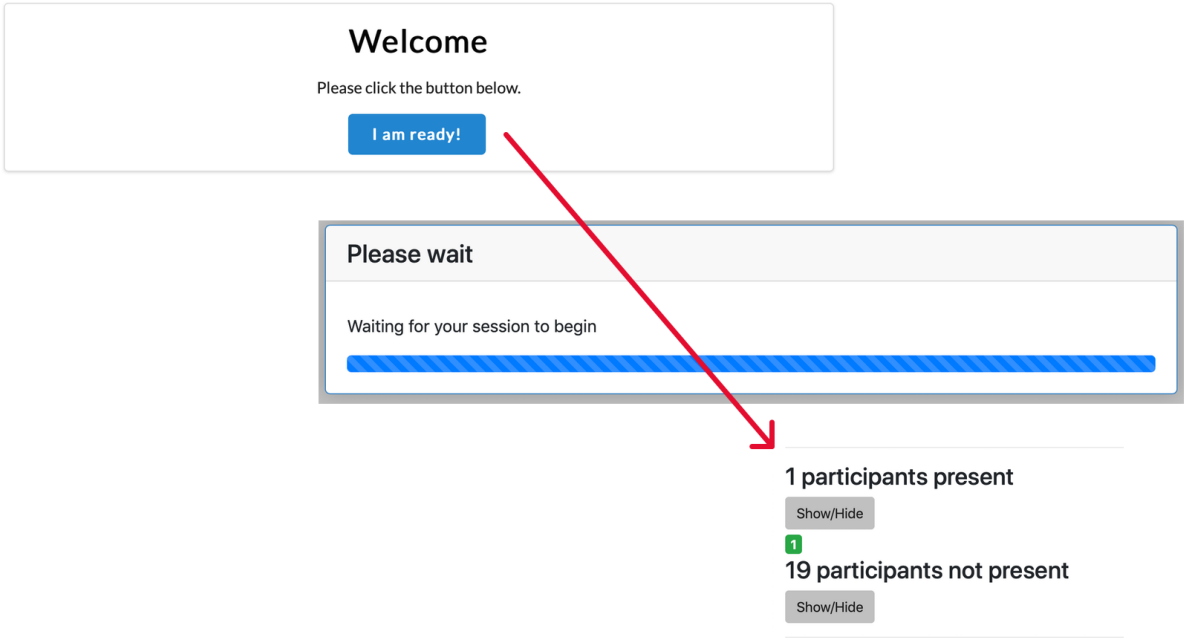
At any given time, experimenters can also monitor the state of their oTree instance. The web interface clearly communicates instance status (green = ready to run experiments / orange = not ready), Git repository URL and credentials, oTree server URL, admin password, and current room setup, as well as the number of web and (timeout-)worker processes ready to handle incoming requests.

Figure 2: Detail view of an oTree instance



Note: This example detail view shows information for instance "otree1". The green icon indicates that it is completely set up and is ready for experiments. The four foldable sections show various details regarding the Git repository, the oTree web interface, the Lobby configuration, and additional experimenter information. On the right, there are buttons for going to the oTree web interface, setting the admin password, restarting the instance, resetting the database, adjusting the scaling (super-users only) and deleting the installation (super-users only).

Figure 3: Lobby feature schematic



Notes: The schematic shows sections of three screens. The top left screen is the welcoming screen, which is shown to participants when they first sit down at their client computer. Once they confirm their readiness by clicking the blue button, they are forwarded to oTree's room waiting page (screen in the middle). Only now does the participant appear in oTree's room administration interfaces as being present (bottom right screen).

2.2. Laboratory managers

Generally, laboratory managers will use so-called super-user accounts on oTree Manager's web interface. Super-user accounts behave like regular experimenter accounts, but come with more rights (and responsibilities). Super-user accounts allow the creation and management of individual user accounts for experimenters as well as oTree server instances. Super-users can create user accounts by providing a name for identification, a user name to login, and an e-mail address for communication. User accounts can also be promoted to super-user accounts such that the respective users have access to all features of oTree Manager. This is especially handy if more than one person manages the laboratory. Of course, user accounts can also be deleted.

Super-users have access and management rights to all oTree instances, irrespective of which user the instance belongs to. In terms of functionality, they can generally do everything a regular user can do and more. Specifically, super-users may change the numbers of web and (timeout-)worker processes that are started for each instance. That is, they can scale-up

individual instances to be able to handle a larger number of simultaneous requests.[7] They can also delete oTree instances.

In its default configuration, oTree Manager automatically sets up sub-domains for the oTree instances. This keeps oTree server URLs short and easy to remember. It also allows using a single SSL certificate (with a wildcard entry for sub-domains) for all oTree instances running on the server. This keeps the effort required to add transport layer encryption to oTree instances to a minimum and enables rapid deployment of oTree instances suitable for integration with Amazon Mechanical Turk.

3. Architecture

3.1. Overview

Behind the scenes, oTree Manager creates a new oTree installation for each instance. Each instance comes with its own production-ready PostgreSQL and Redis database servers and provides Git access for easy deployment. By design, instances are completely independent of each other and thus changes to one instance cannot affect other instances.

On the lowest level, oTree Manager uses Docker to containerize and isolate oTree webservers and databases. Docker is a tool that allows software to be wrapped into standardized units (containers) which include most of the dependencies they need to run. These can easily be distributed, automatically set up, and cleanly separate their contents from each other and the host system. Importantly, containers are much more resource efficient than virtual machines. Each virtual machine comes with its own operating system, kernel, and software libraries. Often, it also has a pre-defined, fixed allocation of hardware resources. Containers, in contrast, run on top of the host's operating system. They can share its kernel, libraries, and hardware resources while maintaining a high level of separation. This reduces overhead and results in quicker start-up times and a higher number of 'guests' that can be run on the same host computer.

oTree Manager relies on Dokku which is a lightweight, open source Platform as a Service (PaaS) implementation. It serves as the fabric between the user-facing web interface and the lower level Docker containers. Dokku manages containers, handles access rights and provides the Git repository functionality. In addition, it provides Heroku compatibility, which simplifies deployment for experimenters, as they do not need to make any changes to their local oTree

[7] This is ultimately limited by the performance of the server oTree Manager is running on as well as the number of instances in use.

development installation before pushing their code into production. At last, Dokku also handles sub-domain creation for the individual instances.

The web interface is built using the Django web framework, which is also the basis for oTree itself. Django is extended by Django Channels to run background processes (i.e. interface with Dokku) and provide near real-time feedback to users via Websocket notifications. Semantic-UI provides consistent and intuitive user interface elements.

3.2. Details

The details of the processes taking place behind the scenes are described best by walking through a standard operation. The operation we take a closer look at is the creation of a new oTree instance with subsequent deployment of an experiment. The process is initiated by a super-user, who provides a name for the instances and assigns it to an existing experimenter account using the web interface. oTree Manager makes sure that instances names are both unique and URL safe.

If these conditions are met, the multiple tasks are sent to Dokku: 1) A new empty app container is created which comes with a Git repository interface and is ready for Heroku-style deployments. 2) Two containers for PostgreSQL and Redis databases are created and linked to the app. 3) The assigned experimenter is granted proper access rights. Each of these steps is run as a separate task in the background using Django Channels. Once these tasks succeed or fail, they trigger notifications, which are shown on the logged-in super-user's interface. To facilitate this, a Websocket connection between the super-user's browser window and the oTree Manager server is kept open. Websockets are an efficient way to allow near real-time communication between server and client without the need for the browser to reload the page or send repeated queries to the server in the background.

Once the background tasks have completed, the new oTree instance shows up in the dashboard as *ready for deployment* (orange icon). Its detail view prominently shows the Git URL which is used to transfer the experiments to the instance. The assigned experimenter can add this URL as a new remote repository to their existing Git repository used during development. The web interface provides a step-by-step guide for the experimenter to follow along. Once the remote is set up, the experimenter can simply push the oTree project to the server instance. As soon as such a push event is picked up by the Dokku app, the oTree container is first re-built from scratch including its dependencies and then deployed.

At this point, if the instance has been deployed successfully, the state of the oTree container switches from *ready for deployment* (orange icon) to *ready for use* (green icon). For easy verification, the detail view now shows the currently deployed Git commit identifier. It also changes the presentation to focus on details of the deployed oTree instance such as its URL and the currently set oTree web interface admin password. Note that it is possible to change the password or adjust the number of processes even if the experiment has not been deployed, yet. Thus, super-users can pre-configure instances for experimenters, if they desire.

Actions such as restarting or deleting the instances, resetting the database, changing the admin password or scaling the number of worker processes work in a very similar way. User interactions trigger Dokku tasks, which are run in the background through Django Channels. These in turn notify the user once they have completed.

A complete manual, which includes user guides for both experimenters and super-users, as well as detailed installation instructions, is available on Read the Docs.[8] The source code of oTree Manager is published under the MIT License and available on GitHub.[9] In the spirit of open source, everyone is invited to contribute to the continued development of oTree Manager and its documentation.

4. Conclusion

oTree Manager makes it easy to set up, run, and manage multiple, production-ready oTree instances on a single machine. It comes with an intuitive web interface, which makes oTree installation management easier for both experimenters and laboratory managers. Instances managed through oTree Manager are completely independent from each other, come pre-configured for production use, and provide a handy Lobby feature for use in experimental laboratories. Using individual Docker containers for separate instances reduces resource requirements compared to traditional virtual machine setups and speeds-up initial deployment.

References

Chen, D. L., Schonger, M., and Wickens, C. (2016) oTree – An open-source platform for laboratory, online, and field experiments. *Journal of Behavioral and Experimental Finance*, 9, 88-97.

[8] http://docs.otree-manager.com
[9] https://github.com/otree-manager/otree_manager

Appendix: Glossary

Cloud-hosted	Typically, web services run on dedicated server hardware in data centers. This is expensive and does not easily scale to growing resource demands. Cloud-hosting runs web services in a virtualized environment which behaves like a single server, but can actually use resources from multiple machines and thus easily scale to growing performance demands. Multiple virtualized servers commonly share hardware resources which increases cost efficiency.
Dependencies	A key principle in software development is "don't repeat yourself" (DRY). Consequently, many software packages rely on other, supporting software to provide their functionality. These other required packages are commonly called dependencies.
Deployment	During its development process, software passes through various stages. These include actual development, testing, and finally deployment. Deploying a software means putting it into real-world use. For software to be considered ready for deployment it typically has to pass testing and quality control and be (mostly) free of known issues.
SSH (keys)	SSH stands for Secure Shell which is a network protocol for encrypted communication over unencrypted connections. It is typically used to log in to remote computers through a command line interface. SSH supports username and password credentials as well as identification and authentication through public-key cryptography. The latter is more secure and more convenient as it does not require the users to create and remember secure passwords.
Django	Django is an open-source web framework written in Python which makes it easy to develop dynamic, data-driven websites. https://www.djangoproject.com
Django-Channels	Django-Channels is an extension of Django, which enables the use of more communication protocols. These are, for example, required for near real-time communication between browsers and web servers. https://channels.readthedocs.io
Docker	Docker bundles software into containers. These containers can be easily distributed and run on top of many common operating systems. Containers bring their own dependencies, tools, and libraries, but share the kernel with the host system. This allows them to be more resource efficient than other virtualization techniques. https://www.docker.com
Dokku	Dokku is a small Platform as a Service implementation. As such it calls itself "mini-Heroku", because it provides a user experience similar to Heroku. At its core, it is a collection of

	software scripts which tie together Git repositories and Docker containers. http://dokku.viewdocs.io/dokku/
Git (repository / remote)	Git is a free and open-source version control system. A repository contains all files of a project which are under version control. Git allows to mirror project repositories across multiple machines. A mirror on a different computer is typically called a "remote". https://git-scm.com
GitHub	GitHub is an online platform which hosts Git repositories. It provides a web interface to the repositories and augments them with useful collaboration features such as issue trackers, wikis, and team management. https://www.github.com
Heroku	Heroku is a cloud Platform as a Service provider. It makes it easy and quick to host websites and web services written in a multitude of languages. It is one of the recommended ways to deploy oTree experiments to production if an experimenter or lab is not running their own dedicated server infrastructure. https://www.heroku.com
Kernel	The Kernel is a central component of each operating system. Almost all input and output requests from software pass through it on their way to the different hardware components.
MIT License	The MIT License allows software to be used for private and commercial purposes, allows it to be distributed, and modified. It limits liability and offers no warranty. A copyright notice as well as the permission statement need to be included if substantial portions of the licensed software are copied or re-used. https://opensource.org/licenses/MIT
Multi-user support	Software can be designed for use by a single user or for use by multiple users. The latter requires some form of user- and credentials management to be included. It is typically expected that software which supports multiple users keeps their work separate to prevent interference and promote data privacy and protection.
Platform as a Service	A Platform as a Service (PaaS) is software which reduces the complexity of developing and running web services. These platforms are typically pre-configured for common deployment patterns and provide easy access to databases and storage providers. Often they run on cloud infrastructure which enables them to scale quickly and transparently to growing resource demands.
Port	Ports are endpoints of computer network communications. Ports can be imagined as doors to the computer. As such ports can be closed and locked or they can be open. Software which communicates over the network "listens" to assigned ports for incoming connections or sends own messages addressed to ports on the receiving end. Ports are associated with an IP

	address and the network protocol to be used. Typically they are appended to the remote address with a colon (e.g. :8000).
PostgreSQL	PostgreSQL is an object-relational database software. It stores data in forms of tables in which columns correspond to variable names and rows to individual entries. The Structured Query Language (SQL) is used to manage data contained in the database, giving it its name. https://www.postgresql.com
Read the Docs	Read the Docs is an open-source software documentation hosting platform. http://readthedocs.org
Redis	Redis is an in-memory key-value database. Data storage is tailored towards fast retrieval. As such, there is no pre-defined structure of data as in typical object-relational databases. Each record can contain its own individual collection of fields. https://redis.io/
SSL / TSL	Secure Sockets Layer (SSL) is the predecessor to Transport Layer Security (TLS). Both are network protocols for encrypted communication and are used to provide privacy and data integrity between clients and servers.
URL	URL stands for Uniform Resource Locator. URLs are more commonly known as web addresses.
Virtual environment	On computer systems used by multiple users one goal is to keep users' data separate from each other and private. While users may share some resources such as installed software, it can be beneficial to also have software separation. This way, each user can keep their own configuration and set of dependencies as well as specific versions, which might otherwise lead to conflicts between users in shared environments. Virtual environments make this separation possible.
Virtual Machine	A virtual machine is a virtualized (simulated) computer running on top of a host's operating system. It typically emulates a complete set of hardware, requiring the installation of its own, separate operating system to be useful. While there is a large degree of separation between different virtual machines running on the same host, they also come at a large resource overhead, because each one brings their own, complete operating system.
Web/worker processes	oTree runs multiple types of processes. Web processes handle incoming network requests and serve pages. Worker processes make sure that timeouts occur, even if a participant in the experiment has closed their browser. Multiple web processes can improve performance if many requests are to be handled simultaneously, for example in large online (Amazon Mechanical Turk) experiments.

Websocket	Websocket is a network communication protocol. It allows sending and receiving messages on the same ports typically in use by web servers. Because the connection between server and client is kept open after initialization, it enables near real-time communication between servers and clients. It can be used to implement chats for example.